BEBR

FACULTY WORKING PAPER NO. 1322

Rule Learning in Knowledge-Based Decision Support Systems: An Integration of Problem Solving and Inductive Inference

Michael J. Shaw

College of Commerce and Business Administration
Bureau of Economic and Business Research
University of Illinois, Urbana-Champaign

BEBR

FACULTY WORKING PAPER NO. 1322

College of Commerce and Business Administration

University of Illinois at Urbana-Champaign

February 1987

Rule Learning in Knowledge-Based Decision Support Systems: An Integration of Problem Solving and Inductive Inference

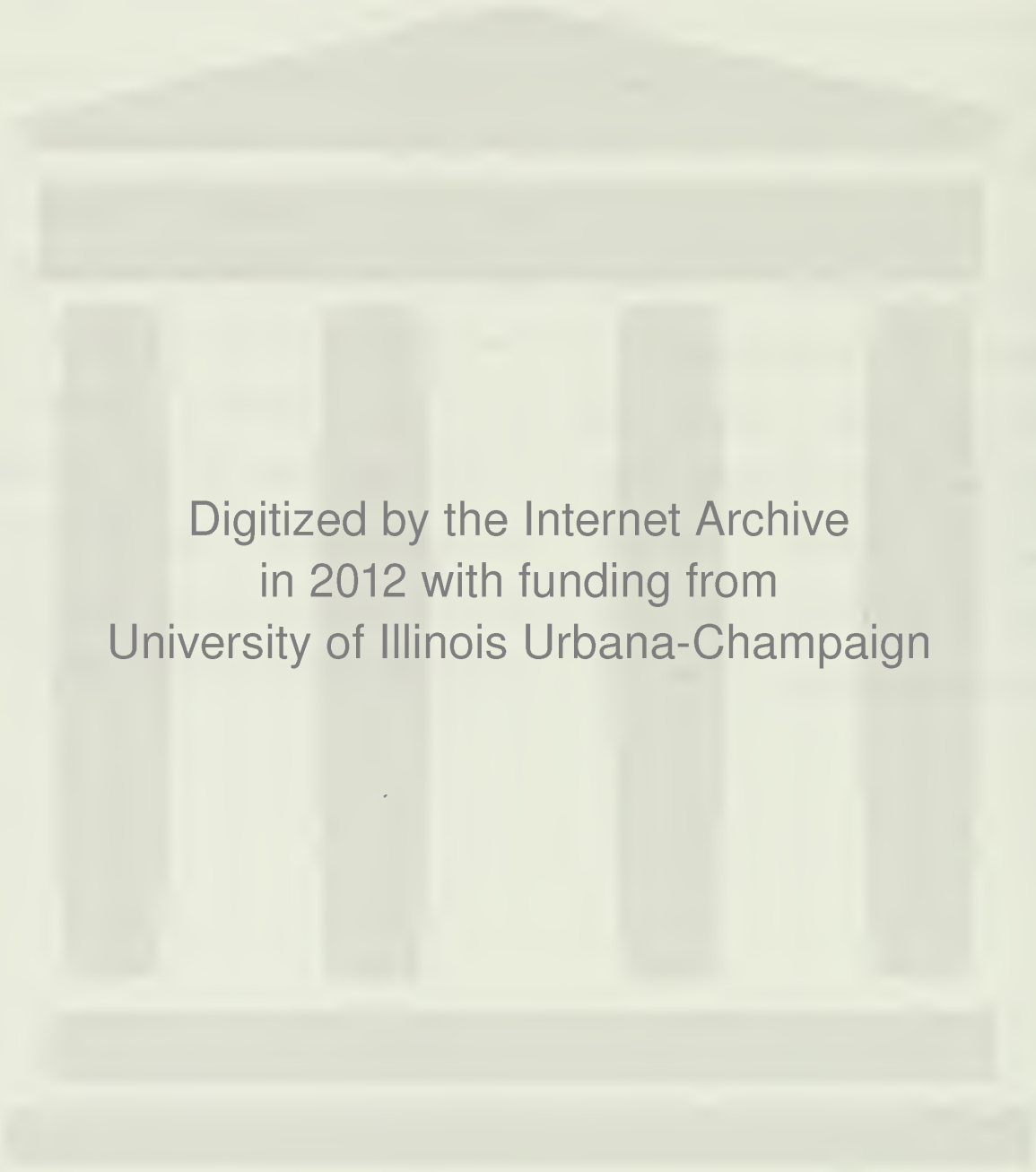
Michael J. Shaw, Assistant Professor
Department of Business Administration

Abstract

This paper is concerned with the methods for incorporating inductive learning in decision-support systems (DSSs) to make the DSS capable of deriving problem-solving knowledge, in the form of rules, by observing the decision examples made by the decision-makers. The intent is to add a new aspect--the learning ability--to the traditional DSSs and make them truly intelligent systems for decision support.

Throughout this paper the application of inductive inference methods for rule learning is addressed based on the application environment of MARBLE, a knowledge-based DSS we have developed for evaluating business loans. We shall describe inductive inference techniques for incrementally learning rules in a hypothesis space and for learning multiple concepts from a set of decision examples. Application examples and an empirical study are presented to demonstrate inductive learning as an enhancement to the effectiveness of knowledge-based DSSs.

Key words: Decision Support Systems; Artificial Intelligence.



1. Introduction

Features such as explanation ability, heuristic inference, reasoning with uncertainty, and structured representation of knowledge make the knowledge-based expert system an effective tool for aiding decision making. Recently there have been emerging research efforts to tailor the knowledge-based expert system methodology to designing decision-support systems (DSSs) (Elam and Henderson [1980], Goul et al. [1985], Stohr [1985]). To stress the decision-support nature of the system and the integration of various decision-support functions in the knowledge-based environment, this type of system is referred to as knowledge-based DSSs.

Most existing DSSs perform decision-support by a deductive problem-solving mechanism: it matches the goals to be achieved against known facts or rules; the invoked rules, in turn, would generate new goals to be satisfied. This rule-selecting process continues until all of the original and intermediate goals are satisfied or until some goals prove to be unsatisfiable (Newell and Simon [1972]). This paper is aimed at adding another dimension to the operational design of knowledge-based DSSs: the ability of the system to learn. Learning has been recognized as an important feature of any intelligent system. There are two aspects in decision-support where learning can come into play: (1) the learning of decision rules for the knowledge base, i.e., the knowledge-acquisition process and (2) the learning of refining existing rules by observing prior problem-solving experience, i.e., the knowledge refinement process. To achieve these learning functions, the knowledge-based DSS must be equipped with an inductive inference

engine complementary to the deductive problem solver. Thus, an important research issue concerns the inductive inference techniques for rule learning and for knowledge acquisition.

Throughout this paper the application of inductive inference methods for rule learning is addressed based on our design experience with MARBLE, a knowledge-based DSS for business loan evaluation. There are practical incentives to introduce inductive learning in a system such as MARBLE. First, an important part of the DSS contains decision rules used by human experts, i.e., experienced loan officers in MARBLE's case, but our experience is that it is not easy to solicit knowledge from the human expert in the form of rules and the process can become very arbitrary. Second, there may not be experts in some domains, or, when there are several experts specializing in the same area, it is often difficult to get their consensus on the best set of rules to use. Third, even when the decision rules have been determined and employed in problem solving, the system still needs to have a means to refine the rules continuously. We shall show that these problems can be resolved by incorporating an inductive-learning component in the knowledge-based DSS.

Although this paper stresses the integration of deductive problem solving and inductive inference, considerably more discussions will be devoted to the latter, since the deductive problem-solving mechanism for performing various types of decision-support have been relatively well discussed in the literature (e.g., Bonczek et al. [1981], Sprague and Carson [1982], and Stohr [1986]). The remainder of the paper is organized as follows. Section 2 reviews the knowledge-based decision-support

environment of MARBLE and describes the problem-solving method it uses. Section 3 discusses inductive inference methods for rule learning and describes a searching algorithm for incremental learning. Section 4 presents an inductive learning method that generates rules from a set of examples; an empirical study is used to validate the decision-support function that can be achieved. Section 5 concludes the paper.

2. Background

2.1 MARBLE: A Testbed for Knowledge-Based Decision Support

The research described in this paper is part of an ongoing effort to develop a knowledge-based decision-support system specializing in financial decision support for commercial banks. The system, referred to as MARBLE (standing for a decision-support system for managing and recommending business loan evaluation), is a MYCIN-based system (Davis [1979] and Buchanan and Shortliffe [1984]) currently consisting of around 80 decision rules for evaluating commercial loans. It applies the judgment exercised by experienced loan officers in arriving at lending decisions for commercial loans.

Typically, the evaluation of a business-loan application is a subjective decision process made independently by loan officers, bank controllers, auditors, and bank examiners. The loan-granting decision usually relies on examining a large amount of historical and pro forma financial information and on such judgmental evaluation as the company's market characteristics, industry performance, management competence, and accuracy of the information obtained. The loan-evaluation decision is traditionally analyzed by statistical linear models, such as regression analysis (Orgler [1970]) or multivariate

discriminant analysis (Kaplan and Dietrich [1982]). As pointed out by Haslem and Longbrake [1972] and Kaplan and Dietrich [1982], statistical analysis with linear models cannot capture the subjective judgments and the qualitative evaluation so important in the lending decision. In essence, the approach used by MARBLE is akin to the heuristic simulation method employed by Cohen, Gilmore, and Singer [1966]; they both simulate the decision process of loan officers. MARBLE, however, employs production rules as the basic knowledge representation, which has been pointed out as an effective model of the human decision-making process (Anderson [1983]). In addition, the recent knowledge-based technology enables MARBLE to be equipped with uncertainty reasoning, explanation, and incremental refinement capabilities. As will be shown, inductive learning can be applied to enhance MARBLE's performance further by automatically acquiring decision rules for loan classification. There are two schools in the attempt to develop metal models for describing learning processes: (1) the connectionist model, which describes mental processes in terms of activation patterns defined over nodes in a highly interconnected network; and (2) the production-system model, which describes mental processes as symbol manipulation in a production system (Anderson [1983] and Newell [1973]). The method we used for incorporating learning in MARBLE essentially takes on the second approach in which learning is achieved by rule-augmenting.

2.2 Integrated Problem Solving in DSS

Characterized by the often large amount of data and program modules (models) involved, a DSS is usually linked with an external database

and a model base (Alter [1980], Bonczek et al. [1980], Sprague and Carson [1982]. The problem solver of MARBLE can be viewed as a production system (Newell [1973] and Davis et al. [1977]) where production rules are used to represent (1) procedural knowledge, (2) decision heuristics, and (3) model abstraction. Procedural knowledge is the knowledge about the essential steps, mostly related to information collection, for making a given decision; for example, to evaluate a company's credit-worthiness, the necessary supporting information includes the competence of the management, the outside credit rating of the firm, and credit analysis on the firm's financial data. This piece of procedural knowledge is shown as Rule 073 in Appendix A. The decision heuristics are rules of thumb used by loan officers. Because of the inherently judgmental nature, this type of rules needs considerably more effort to obtain and refine. The rules generated by inductive learning belong to this category. The third type of rules is used to represent the model knowledge available for decision support; these rules indicate the application requirements of each model and the precedence relations between models.

With these different types of decision-support knowledge, the problem solver serves as a bridge linking the decision maker's problem environment with the appropriate models, data, and decision rules residing in the DSS. An example of the consultation session performed by the problem solver interacting with the user is shown in Appendix 1.

The basic inference mechanism for accomplishing decision support tasks in a knowledge-based DSS, such as MARBLE, is based on the problem-solving theory established by Newell and Simon [1972], which treats

problem solving as a process of search through state space. A problem is defined by the initial state, the desirable goal state, a set of operators that can transform one state into another, and constraints that an acceptable solution must meet. Problem-solving under this theory involves the selection of appropriate sequence of operators that will succeed in transforming an initial state into a goal state through a series of steps. For decision-support tasks, these steps selected in the process are primarily information processing activities, resulting in a plan of action. Utilizing information from the knowledge-base, external database, dynamic database (sometimes referred to as black-board), and model base (Bonczek et al. [1981, 1983], Dolk and Konsynski [1984] and Dutta and Basu [1984]). In the case of MARBLE, the model-base can contain program modules for financial analysis, mathematic pro-gramming routines, forecasting, simulation, or regression algorithms. The external database typically contains the historical loan data and financial information of companies applying loans. Therefore, special care must be taken to handle the interface between the system's knowledge-base, model base, and database (Shaw and Tu [1985]).

3. Inductive Inference for Rule Learning

3.1 Decision-Support Framework with Machine Learning Capabilities

The ability to learn has long been recognized as an essential feature of any intelligence system. Dietterich et al. [1981] cate-gorizes learning methods into four areas based on their behavioral characteristics: rote learning, learning by being told, learning from examples, and learning by analogy. Most existing DSSs use learning by being told for acquiring problem-solving knowledge; they take the domain

knowledge from experienced decision makers in the field (e.g., experienced loan officers in the case of MARBLE) and transform the knowledge into the representation form in the knowledge-base of the DSS. This type of learning, as a form of knowledge acquisition, is shown as (b) in Figure 3.1.

Insert Figure 3.1 Here

Machine learning can be incorporated in DSS in three different forms:

(1) Incremental learning, in which rules are progressively constructed and refined based on new observations;

(2) Learning from an example set, in which decision rules are derived based on a given set of positive and negative examples; and

(3) Rule modification, in which the rules in the knowledge-base are modified so as to improve the performance of the problem solver.

The difference between the first and the second types is that incremental learning constructs rules progressively based on new input examples whereas learning from an example set would not accept any new examples after the rules are constructed; moreover, the latter is concerned with multiple-concept learning. The third type of learning, concerning rule-modification, identifies faulty decisions made by the system and uses them to modify the content of the rules. The discussions in this paper will focus on the first two aspects of learning, shown as (a) in Figure 3.1. Although rule-modification is beyond the scope of this paper, it can be shown that (e.g., Bundy et al. [1985]) this type of learning can be used for incremental learning, in which the problem-solving traces serve as the input examples; a problem-solving trace is a sequence of steps represented by instantiated

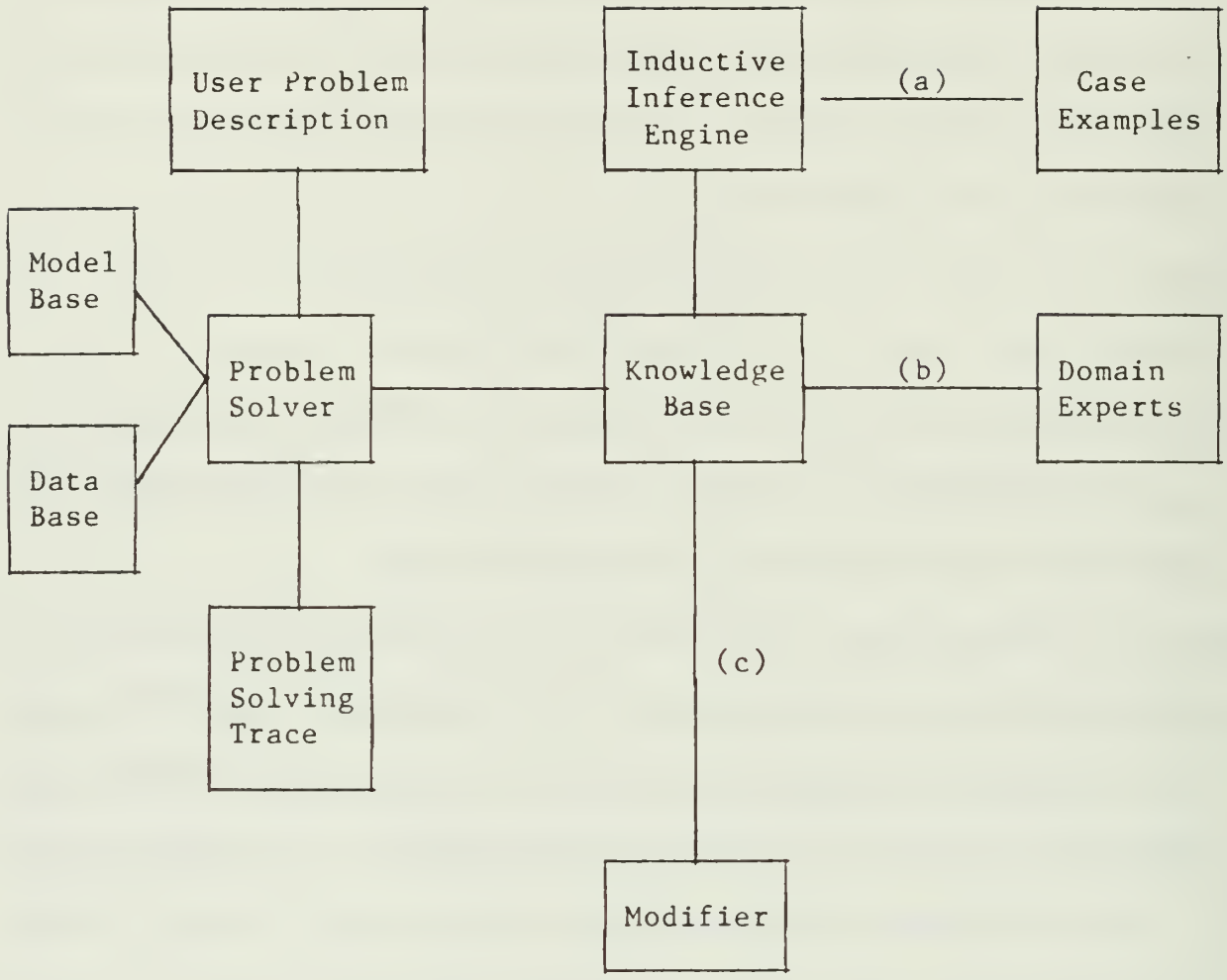


Figure 3.1. Integration of Problem Solving and Inductive Inference Process

rules. A modifier will change the content of rules to make them consistent with those examples. This is shown as (c) in Figure 3.1.

3.2 Inductive Inference

Inductive learning can be defined as the process of inferring the description of a class from the description of some individual objects of the class. Learning examples are given in the form of instances which are described by a vector of attribute values. Each class can be viewed as a concept which is described by a concept recognition rule as a result of inductive learning; if an input data instance satisfies this rule, then it represents the given concept. A concept is a symbolic description expressed in some description language, that is TRUE when applied to a positive instance and FALSE when applied to a negative instance of the concept (Winston [1975]). For example, a recognition rule for the concept "class A firm" might be:

> "A firm whose asset exceeds $1,000,000.00, total debt is less than $250,000.00, and whose annual growth rate is more than 10%."

Using first-order predicate calculus (FOPC) as the knowledge representation, such a concept can be represented by well-formed-formulas (WFFs) in first-order predicate calculus (Bonczek et al. [1981], Vere [1975]). In this example the concept can be represented by a conjunction of attribute descriptions:

$$\text{customer}(t) \wedge (\text{asset}(t) > 1{,}000{,}000) \wedge (\text{total-debt}(t) < \$250{,}000) \wedge (\text{AGR}(t) > 0.10) \rightarrow (\text{class}(t) = \text{'A'})$$

An alternative way to represent such a concept is to use the variable-valued logic (VL) proposed by Michalski [1983]. The VL

language is an extended form of if-then rules where many-valued logic proportional calculus is involved. The premise section of each rule is a conjunction of multivalued attribute variables; each attribute is enclosed by a bracket with the corresponding attribute values. A selector relates an attribute to a value or a disjunction of values. For example, [type-of-firm = manufacturing v steel] is a selector which assigns two disjunctive values to an attribute. The conjunction of such selector forms a complex. The aforementioned concept recognition rule can be represented by the VL formalism as follows:

[assets > $1,000,000][total-debt < $250,000]

[AGR > 0.10] → [class : 'A'].

(Note that the LHS of the rule is a conjunction of selectors.)

The set of data examples F can be viewed as a collection of implications

(A) $\quad F: \{e_k^i \rightarrow K_i\}, \; i \in I,$

where e_k^i (a training example) denotes the k^{th} example of a concept (class) asserted by predicate K_i; and I is the set of class indexes. It is assumed that any given example represents only one concept. The inductive hypothesis H can, then, be characterized as a set of concept (class) recognition rules:

(B) $\quad H: \{D_i \rightarrow K_i\}, \; i \in I,$

where D_i is a concept description of class K_i; that is, when an input data instance satisifies the conditions described by D_i, the instance

is considered to be in class K_i. (Note: we use "=>" to denote logical implications and "→" to denote classification assignments.)

Inductive learning is essentially an inference process which generates D_i for each class K_i to describe every individual instance in that class. Let E_i, $i \varepsilon I$, be a description satisified by all training examples of class K_i, i.e., $E_i = \bigvee_k e_k^i$, then

(C) for all $i \varepsilon I$: $(E_i \Rightarrow D_i)$.

To satisfy formula (C), the inductive description of each class is typically a conjunction of some simple attribute values shared by all instances in the class. For the classification application, the primary goal is to distinguish a given class from a fixed number of other classes. Therefore, a classification recognition rule is determined by obtaining the "maximally specific conjunction" (Vere [1975], Dietterich et al. [1981], and Michalski [1983]) of the training cases in that class; that is, using the set of conjunctive terms which satisfy the condition represented in (C) as well as the following condition:

(D) for any $i, j \varepsilon I$ $(D_i \Rightarrow \sim E_j)$, if $j \neq i$.

In other words, if the given data case satisfies a description of some class, then it cannot be a member of the training set of any other class. For a class i, the training examples of that class are referred to as positive examples, the training examples of all other classes are negative examples. Formula (C) is referred to as the completeness condition and formula (D) is referred to as the consistency condition. To satisfy the goal of inductive learning--i.e., to derive a decision rule

for each class that covers all the training examples of one class but none of the training examples of any other classes--a concept recognition rule is acceptable only if it achieves both the completeness and consistency conditions.

3.3 Inductive Inference as Search

The process of inductive inference is itself a problem-solving process where solutions, the inductive concept descriptions, can be obtained through searching (Lenat [1976], Quinlan [1979], Rendall [1983]). Concept descriptions are derived through a sequence of transformations to generate the goal descriptions. The states are defined by the possible symbolic concept description, structured in a search space called the hypothesis space. Based on this paradigm, the inductive inference system consists of these components: (1) the hypothesis space which organizes all the concept descriptions by a partial ordering; (2) the class of transformation rules being considered, such as the generalization rules; (3) the set of training examples; and (4) the criteria for a successful inference, such as the simplicity of hypothesis generated, goodness of fit, completeness, and consistency (Anglin and Smith [1983]).

Generalization is essential for making inductive inference. If a concept description Q is more general than the concept description P, then the transformation P to Q is called generalization. More formally, the degree of generality/specificity can be defined as follows:

Definition 3.1

If P and Q are two concept descriptions, P is more general than Q if and only if there are more instances covered by P than by Q. That is, $\{\omega \varepsilon W | C(P,\omega)\} \geq \{\omega \varepsilon W | C(Q,\omega)\}$, where W is the set of all possible instances, and C is the cover predicate (C(u,v) = TRUE if u covers v).

The most common way to perform generalization is to relax the constant of a description to a substitutable variable. For example, the following instance, consisting of three attributes, describes the financial characterization of a firm:

e^i: {(manufacturing, medium-debt, large-equity)}

A possible generalization of e^i is to relax one of the attribute values to a substitutable variable, represented by ?x:

{(manufacturing, ?x, large-equity)}.

Based on the concept of generalization, inductive inference can be viewed as a process of generalizing the initial descriptions as observed from the examples, and intermediate concept-descriptions until the inductive concept-descriptions consistent with all examples are found. Thus, the generalization relations between concept-descriptions provide the basic structure to guide the search in inductive inference. This generalization relation can be accounted for in inductive inference by ordering concept-descriptions according to their degree of generality/specificity and by using transforming rules

to achieve generalization. Since there are usually multiple ways to generalize a concept description, the generalization relationship provides a partially ordered structure for the hypothesis space. The inductive inference algorithms we shall subsequently describe take advantages of this structure to expedite the search. Alternatively, the searching process can also be guided by some information theoretical measure (Lee and Ray [1986] and Quilan [1979]).

Given a set of positive and negative examples, the inductive concept description should be consistent with all the positive examples whereas not matching with any of the negative examples. In terms of the degree of generality/specificity, the inductive description should be (1) as general as possible, but only to the degree that it would not cover any of the negative examples, (2) as specific as possible, but it should at the same time be general enough to cover all the positive examples. Thus, the search space within which inductive inference is executed is bounded by two sets: (1) the set of maximally general generalizations, denoted by G, and (2) the set of maximally specific generalizations, denoted by S. G and S are defined as follows:

G = {g|g is consistent with the observed instances, and there is no generalization which is both more general than g, and consistent with the instances}; and

S = {s|s is a generalization that is consistent with the observed instances, and there is no generalization which is both more specific than s and consistent with the observed instances}.

Based on the partially ordered hypothesis space and the bounds provided by the G set and the S set, the following algorithm (Mitchell [1982]) can be used to construct inductive rules by sequentially inputing training examples.

The Candidate-Elimination Algorithm

Initialize the G set to contain only null descriptions, and the S set to contain the first positive example.

Do While G ≠ S

Begin

Accept a new training example, e^i

Begin

If e^i is a positive example,

Then Begin

(1) Remove from G all concepts that do not cover e^i.

(2) Update S to contain all of the maximally specific generalizations of e^i while generalizing the elements in S as little as possible.

End

Else If e^i is a negative example,

Then Begin

(1) Remove from S all concepts that cover e^i.

(2) Update G, as little as possible, so that its elements would not cover e^i.

End

End

End

	Observations			Type
1.	{(manufacturing	small-liability	large-inventory)	positive
	(software	large-liability	large-A/R)}	
2.	{(manufacturing	large-liability	large-A/R)	positive
	(software	small-liability	large-inventory)}	
3.	{(manufacturing	large-liability	large-inventory)	negative
	(software	large-liability	large-inventory)}	

Figure 3.2 Observations of the Learning Example*

*Each example is presented in the following form:

(type-of-industry liability-level equity-characteristics)

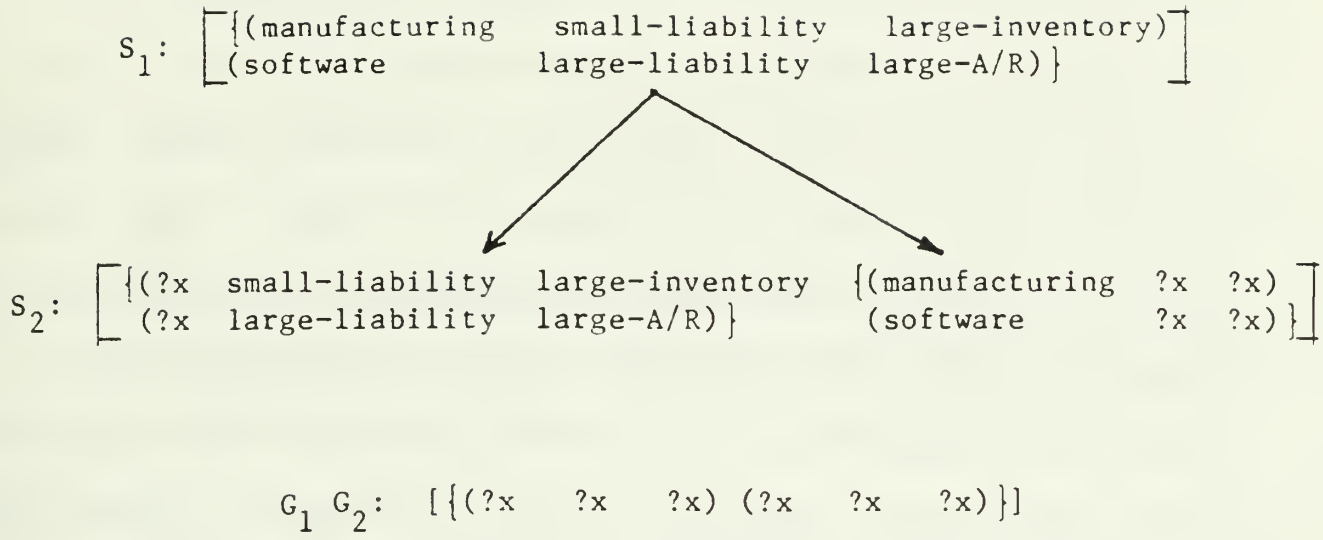


Figure 3.3 An Intermediate Searching State
After the First Two Observations

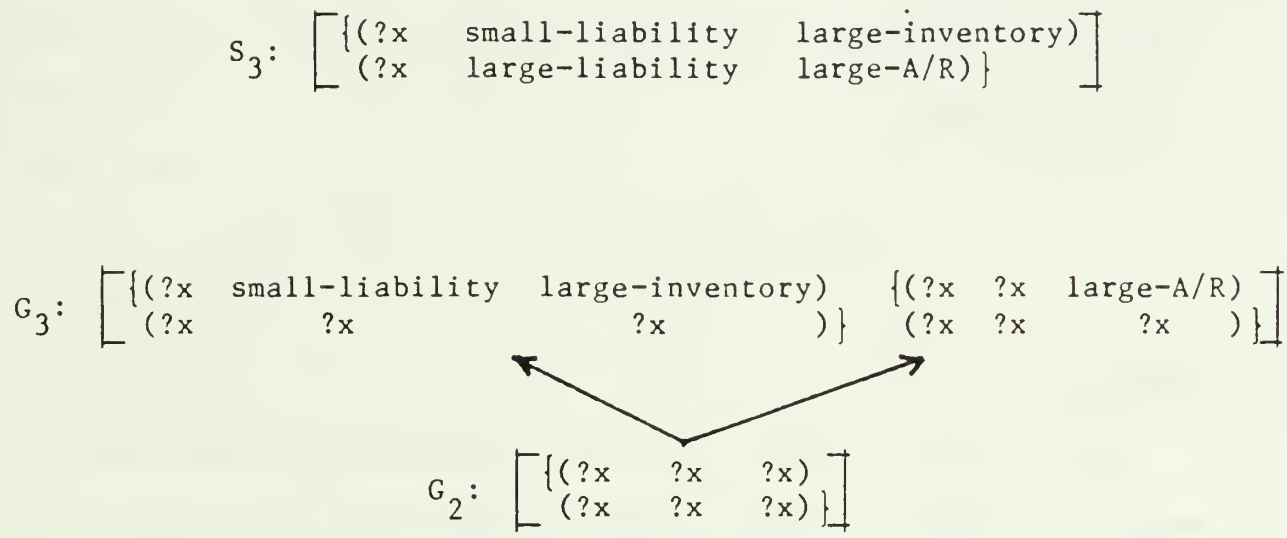


Figure 3.4 A New State After the Third Observation

Since the hypothesis space is partially ordered by the generalization relation, the sets Q and S serve as bounds that confine the possible inductive concept descriptions in the search space. Positive examples force the algorithm to generalize, in order to cover the new example, and remove those concept descriptions that are not general enough from the G set. Conversely, negative examples force the algorithm to specialize, so as to avoid covering the new example, and remove those concept descriptions that are too general from the S set. The search space gradually shrinks in this manner until both the G and the S sets converge to the desired concept description.

For example, suppose that each training example characterizes a firm and its major subsidiary by three attributes: type-of-company, level-of-liability, and type-of-equity. Three such examples, e^1, e^2, and e^3, are shown in Figure 3.2, where e^1 and e^2 are positive examples; e^3 is a negative example. If e^1, e^2, and e^3 are input to the Candidate-Elimination Algorithm, the G and S sets are shown in Figures 3.3 and 3.4 progressively.

Insert Figures 3.2, 3.3 and 3.4 Here

Lemma 3.1

Let N_P be the number of positive examples, N_N be the number of negative examples, s be the largest number of element in the S set, g be the largest number of element in the G set. The computation time of the Candidate-Elimination Algorithm is bounded by $O(sg(N_P+N_N) + s^2N_P + g^2N_N)$; its memory requirement is bounded by $O(s+g)$.

Proof:

(i) Computation complexity: for each input example that is positive, e^k, three types of operations are performed: (a) compare e^k with each element in the G set, (b) compare e^k with each element in the S set, and (c) compare every element in S with each of the other elements in S. The resulting computation-time bound is $O(g \cdot N_P + sN_P + s^2 \cdot N_P)$. Similarly, the computation-time bound for processing the set of negative example is $O(s \cdot N_N + gN_N + g^2 N_N)$, thus a total of $O(sg(N_P+N_N) + s^2 N_P + g^2 N_N)$.

(ii) Storage requirement: the algorithm only needs to store the S set and the G set, thus the storage requirement is bounded by $O(s+g)$.

The sets G and S in the Candidate-Elimination Algorithm represent the hypothesis space in an efficient manner, summarizing the information gathered from the observed training examples. As a result, the storage requirement is small. However, a generalized concept description satisfying all the training examples is found only when the G set and the S set converge. Problems arise when the given set of training examples is not sufficient to make the G set and the S set equal. In this case the algorithm results in a partially learned inductive rule and may cause ambiguity in classifying instances. In the next section we shall describe an inductive inference algorithm for learning multiple concepts; that algorithm would always generate unambiguous rules from the given set of examples.

4. Rule Learning From a Set of Examples

4.1 An Inductive Learning Algorithm

The Candidate-Elimination Algorithm is for the learning situations where training examples are input sequentially; rules are constructed

accordingly by adjusting the G set and the S set based on the examples observed. Such a sequential learning method is especially suitable for incremental learning where new training examples are not all available initially. For learning tasks in the loan-evaluation domain in which MARBLE is functioning, training examples are usually available as a whole set of data. Thus, an inductive inference method is needed which can take into account all of the training examples at the same time and can guarantee convergence. An additional benefit of using an induction method that considers the training examples all at once as opposed to such a sequential method is that the former usually can handle noisy data among training examples better than sequential learning.

In this section we shall present an Inductive Inference Algorithm for learning multiple concepts from examples. The algorithm is similar to the star methodology of Michalski [1983], in which negative examples are used to constrain the search space. In other words, the algorithm uses negative examples to ensure that the inference process would only search through those consistent descriptions in the hypothesis space. First, the problem of inductive inference can be formalized by the following transformation:

Problem 4.1: (Inductive Inference for Learning Multiple Concepts)

INPUT:

(1) A set of classification examples

(2) Rules of generalization and other domain-specific knowledge

(3) Preference criterion

OUTPUT:

(1) A set of decision rules consisting of inductive concept-description for each of the classes.

The process of inductive learning for multiple concepts begins with the separation of positive and negative decision examples among the whole set of training examples. Letting the set of positive examples be S_P and the set of negative examples be S_N, the goal of the algorithm is then to determine a conjunction of attribute values as the concept description that satisfies the completeness condition, the consistency condition, and the criterion of the induction. The Inductive Inference Algorithm for problem solving can be described as follows:

The Inductive Inference Algorithm

0. Initialize the inductive concept description, D, to be an empty set; read in Sp and SN.
1. Do While not every example is covered by D.

   Begin

   2. Select an uncovered positive example, e^j.
   3. Generate the discriminant of e^j against every element in S_N, resulting in d^{jk}, k=1, ..., N_N.
   4. Generate all of the consistent complexes from d^{jk}, k=1, ..., N_N.
   5. Select the best complex, C_*^j, from 4.
   6. Remove every e^ℓ from Sp if e^ℓ in Sp is covered by C_*^j.
   7. If C_*^j is not already in D

      Then Begin

8. Add C_*^j as another conjunctive element of D(i.e., $D \leftarrow DVC_*^j$).

End

End

The learning program would iteratively choose an element e^j in S_P and, for every element f^k in S_N, generate a discriminant $d(e^j/f^k)$, or d^{jk}, in Step 3. Mathematically, let

$$e^j = a_1^j \cdot a_2^j \cdot \ldots \cdot a_i^j = \bigwedge_{i=1,n} a_i^j \text{ and}$$

$$f^k = f_1^j \cdot f_2^k \cdot \ldots \cdot f_i^k = \bigwedge_{i=1,n} f_i^k ,$$

where a_i^j and f_i^j are attributes in e^j and f^k, respectively. Then $d^{jk} = d(e^j/f^k) = \bigwedge_{i \in Q} a_i^j$, where $Q = \{i : a_i^j \neq f_i^k\}$ and $d^{jk} = \bigwedge_{i=1,\ell_{jk}} d_i^{jk}$. That is, $d(e^j/f^k)$ is a conjunction of attribute values that can be described by e^j but not f^k.

In Step 4 the program will generate a set of all <u>consistent complexes</u> C^j associated with e^j; each element C_i^j in C^j is a conjunction of attributes not described by any of the negative examples in S_N. Algorithmically, C_i^j is generated by taking an attribute out of each d^{jk}, for k = 1,...n, and forming a conjunction, that is,

$$C^j = \{(\bigwedge_{\ell=1,n} C_{i\ell}^j) : C_{i1}^j \in d_1^{jk},\ C_{i2}^j \in d_2^{jk},\ \ldots,\ C_{in}^j \in d_n^{jk}\}.$$

Thus, there are a total of $\ell_{j1} \cdot \ell_{j2} \cdots \ell_{jn} = \prod_{k=1,n} \ell_{jk}$ consistent complexes for e^j (remember that ℓ_{jk} is the number of attributes contained in the discriminant d^{jk}). Each of the complexes is consistent, since it does not cover any negative example.

The induction criterion should then be applied in Step 5 to choose the best complex C_*^j in C_i^j's based on a utility function,

$$f(C_i^j) = w_1P_1 + w_2P_2 + \ldots + w_rP_r,$$

where P_1, P_2, ..., P_r are the induction criteria chosen by the user, w_1, w_2, ..., w_r show the degree of importance the user gives to the preference criterion. C_*^j is chosen by selecting the highest $f(\cdot)$ value.

For example, the induction criterion--"to satisfy as many positive examples as possible while not covering any of the negative examples"--can be translated to the utility function,

$$\text{MAX } f(c_i^j) = N_P - W*N_N,$$

where N_P is the number of positive examples satisfied by C_i^j and N_N is the number of negative examples that can describe C_i^j. W is a very large number used to discourage N_N from taking any positive value.

In Step 6, positive examples covered by C_*^j will be removed from S_P, and the same procedure will be applied to the remaining S_P and the original S_N again until all positive examples, e^js, are covered. All the complexes thus produced will be combined to form a complete disjunctive description, D, which forms the inductive concept description. This process continues until every positive example is covered by D.

Lemma 4.1

The aforementioned inductive inference algorithm for learning multiple concepts has a computation-time bound of $O(N_p(N_N+n^{N_N}+d))$; the storage requirement is bounded by $O(n^{N_N})$(d is largest number of

complexes in the inductive concept description D; n is the number of attributes in each example).

Proof:

The algorithm consists of four major operations for each positive example:

(a) Generate N_N discriminants; this takes N_N steps.

(b) Generate all the consistent complexes out of the N_N discriminants; this takes a maximum of n^{N_N} steps where n is the number of attribute of each instance.

(c) Select the best complex; this takes as many steps as (b).

(d) Add the new complex to D; this takes a maximum d steps to remove the redundant complexes, where d is the maximum number of complexes in D.

The computation-time bound is thus $O(N_p(N_N+n^{N_N}+d))$. Since the algorithm in each iteration needs to store all the consistent complexes for selecting the best one, the storage requirement is bounded by $O(n^{N_N})$.

The most costly operation in the Inductive Inference Algorithm is Step 3: the generation of all consistent complexes. To resolve this problem, we can use a beam-search method to cut down the size of the search tree. Beam search is a heuristic searching technique first used in HARPY, a speach recognition system (Lowerre [1976]). The idea is to search a number of potentially optimal paths in parallel so as to expedite the search by reducing backtracking nodes. Ow [1984] applied the problem to scheduling. Michalski [1983] referred to this heuristic method as bounded star. The beamsearch version of the algorithm, which is used for the empirical study in Section 4.3, has

the modification that only γ consistent complexes in step 3 would be considered, where γ is the beam width.

Theorem 4.1.

The rules generated by the Inductive Inference Algorithm satisfy both the completeness and the consistency conditions defined in Section 3.1.

Proof:

(i) Completeness:

The algorithm is complete because it uses steps 2-5 to generate new complexes to describe uncovered positive examples until every positive example is covered by the disjunctive concept description, D, produced in Step 8. Thus, D must cover the descriptions contained in any positive example.

(ii) Consistency:

The algorithm is consistent because the final concept description is generated based on the discriminant, $d(e^j/f^k)$, in step 3. Any description covered by the negative examples are excluded in this step. Thus, none of the negative examples are covered by D in step 8.

4.2 An Example: Applying Inductive Learning in the MARBLE System

We shall use the loan evaluation as an example to illustrate the application of inductive learning in MARBLE. The objective is to determine the risk classification of commercial bank loans. In order to describe the default risk on a given commercial loan, a bank usually would use a five-category classification scheme (Kaplan and Dietrich

Name	Description	Type
F1	the rating of management competence	nominal domain = {high, average marginal, reject}
F2	the outside credit rating	nominal domain = {high, average marginal, reject}
Current-assets	using the pro forma balance sheet, the amount of current current assets	linear
Net-worth	the amount of net worth	linear
Total-debt	the amount of total debt	linear
Funds	the funds for debt service to funds provided operations (three years average)	linear
Cash	the amount of cash	linear
Current-liabilities	the amount of current current liabilities	linear
Current-inventory	the amount of current inventory	linear
Average-inventory	the amount of three years average inventory	linear
Avg-profits	three-year average of net profits	linear
Past-account-evaluation	the evaluation of past account	structured
Customer-status	the applicant's customer status with the bank	nominal domain = {current,new}
Account-type	the applicant's account type either in this bank or from from other banks	structured

Figure 4.1 Relevant Attributes for Credit Rating

[1982]). Here, for the sake of simplicity, only three classes, represented by I, IA, II, are actually used in the set of training examples. There are a total of nine training examples: customers A, B, C for class I; D, E, F for class IA; and G, H, I for class II.

An initial set of attributes using historical and pro forma financial information are selected to be included in each input data instance as training examples. As shown in Figure 4.1, this set of attributes includes nominal, linear, and structured attributes. In the more traditional data analysis techniques, such as regression or discriminant analysis, only linear and nominal attributes can be considered. The ability to process structural information constitutes one of the advantages of symbolic processing (as characterized by most AI programs) over numerical calculation (as characterized by statistical analysis). The domain of each structured attribute usually can be represented by a hierarchy of attribute values, corresponding to a generalization tree. Two structured attributes used in this example are shown in Figure 4.2. The tree structure will be used to apply appropriate generalization rules in the induction process.

(i) Training examples.

After choosing the relevant attributes, a set of data descriptions $\{e_k^i\}$, i = 1, 2, 3, and the corresponding class $\{K_i\}$. (K_i = I, K_2 = IA, K_3 = II) are used as training examples (Figure 4.3).

Insert Figures 4.1, 4.2, and 4.3 here

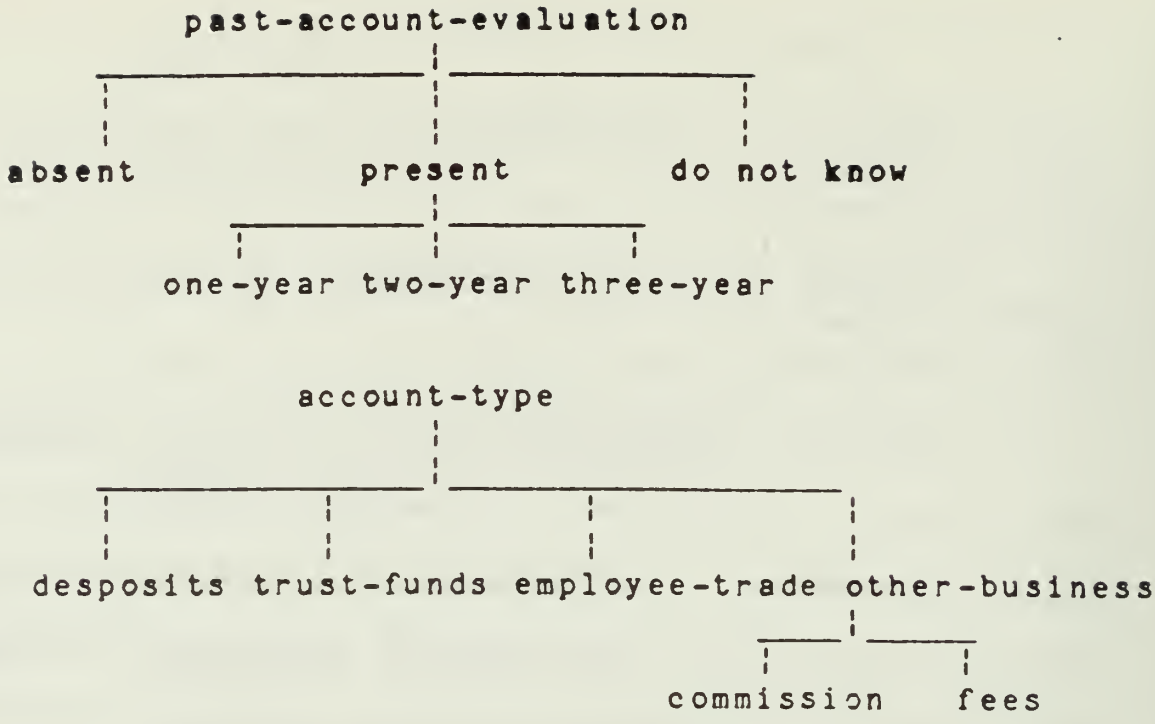


Figure 4.2 Examples of Structured Attributes

	A	B	C	D	E	F	G	H	I
F1	H	H	H	A	H	A	A	M	A
F2	H	H	A	A	A	A	M	A	A
Current assets	57	39	43	42	38	52	45	37	46
Net worth	57	55	49	37	46	40	38	29	36
Total debt	23	17	20	19	28	25	36	27	35
Funds	9	8	7	8	9	6	-9	7	5
Cash	4	3	5	6	4	5	6	6	5
Cur. Liability	39	28	47	55	39	45	57	53	57
Inventory	21	15	18	12	14	11	7	13	14
Avg inventory	9	14	11	6	6	5	3	5	6
Avg-profits	12	15	13	8	9	9	9	8	-0.8
Past-acc-eval	1Y	2Y	3Y	2Y	1Y	1Y	3Y	2Y	NA
Cust-status	C	C	N	C	C	N	N	C	C
Account-type	C	E	D	D	T	E	E	T	T

Figure 4.3 Data of 9 Customers
(all figures in $1,000)

(ii) Generalization rules.

The domain-specific knowledge, represented by the generalization trees in Figure 4.2, can be specified by the following generalization rules:

R1:

[past-account-eval = one-year]V[past-account-eval = two-year]V

[past-account-eval = three-year] --> [past-account-eval = present];

R2:

[account-type = commission]V[account-type = fees]

--> [account-type = other-businesses].

In addition, there are a set of generalization rules, independent of the application, that can be applied in the inductive inference process. Michalski [1983] comprehensively surveyed the generalization rules for transforming and generalizing descriptions. The inductive inference example in Appendix 3 contains generalization rules based on the attributes' generalization tree and based on such rules as the closing-interval rule and the dropping-condition rule.

(iii) Induction criterion

The induction criteria are (1) to maximize the number of positive examples covered, while not covering any of the negative examples, and (2) to include the least number of attributes.

(iv) The inference process and the rules learned

The execution of the foregoing inductive learning algorithm is shown in Appendix 3. The step-by-step execution of the inductive inference algorithm is shown in Appendix 3. The inductive rules for describing the three classes of input are as follows:

1. [avg-inventory $\geq$ \$7,000][net-worth $\geq$ \$47,000] —> [class = I].

2. [\$37,000 $\leq$ net-worth $\leq$ \$48,000][inventory > \$8,000] --> [class = IA].

3. [F1 = H,A][total-debt $\geq$ \$26,000] --> [class = II].

The resulting three decision rules generated are then stored in MARBLE, using the rule format described in Appendix 2, as follows:

```
1. PREMISE:  ($AND (GREATEQ* (VAL1 CNTXT AVG-INVENTORY) 7,000)
                   (GREATEQ* (VAL1 CNTXT NET-WORTH) 47,000));
   ACTION:   (DO-ALL (CONCLUDE CNTXT CLASS-TYPE I TALLY 1000))
2. PREMISE:  ($AND (BETWEEN* (VAL1 CNTXT NET-WORTH) 37,000 48,000)
                   (GREATEQ* (VAL1 CNTXT INVENTORY) 8,000))
   ACTION:   (DO-ALL (CONCLUDE CNTXT CLASS-TYPE IA TALLY 1000)); and
3. PREMISE:  ($AND ($OR ($AND (SAME CNTXT F1 H)
                              (SAME CNTXT F1 A)))
                   (GREATEQ* (VAL1 CNTXT TOTAL-DEDT) 26,000))
   ACTION:   (DO-ALL (CONCLUDE CNTXT CLASS-TYPE II TALLY 1000)).
```

4.3 Empirical Study on a MARBLE Application

To test the performance of the inductive inference method for rule learning in the domain of loan evaluation and risk analysis, we have conducted an empirical study using real-world data. This study concerns using financial data for bankruptcy prediction; the task for the inductive inference engine is to perform concept learning about the characteristics of bankruptcy firms. The learned rules based on such data are used as part of the risk analysis.

The data are obtained from the bankruptcy study conducted by Gentry et al. [1985]. The Standard and Poor's Compustat 1981 Industrial

	Total Number of Testing Cases	Number of Correct Prediction	Percentage Correct
Failed Firms (Positive Examples)	15	11	73.3%
Nonfailed Firms (Negative Examples)	15	11	73.3%

Table 4.1 The Prediction Accuracy of the Inductive Inference Algorithm Using Holdout Sample

	Total Number of Testing Cases	Number of Correct Prediction	Percentage Correct
Failed Firms (Positive Examples)	29	25	86.2%
Nonfailed Firms (Negative Examples)	29	25	86.2%

Table 4.2 The Classification Accuracy of the Inductive Inference Algorithm Using the Whole Sample

Annual Research File of companies, and the Compustat Industrial Files were used to determine companies that failed during the period 1970-81. Balance sheet and income statement information for the failed companies was used to determine the funds flow components (Appendix 4). There were a total of 29 companies of which the complete financial statement information for the year before the failure date was available. These companies are used as positive examples. Furthermore, each of the 29 failed companies was matched with a nonfailed company in the same industry, based on asset size and sales for the fiscal year before bankruptcy. The same set of financial data are provided for each of these nonfailed companies, which serve as negative examples of the concept. The objective of the analysis is to determine whether the inductive inference engine can effectively discriminate between failed and nonfailed companies by the financial data available. The rule learning program, based on the Induction Inference Algorithm, is written in PASCAL on PDP 11/780.

The set of training examples are primarily the funds flow components, listed in Appendix 4, of the failed and nonfailed firms. To test the prediction accuracy of the rules generated by the Inductive Inference Algorithm, we use the holdout sample technique and use half of the sample for rule learning; the rules are then tested on the remainder of the sample. The selection of training examples out of the set of data is based on a degree of representativeness of each data case. Based on the "outstanding representatives" method described in Michalski and Larson [1978], the training examples are selected so that they are most distant from each other. The selected

examples are in a sense the most representative examples. The beam search version of the Inductive Inference Algorithm is used with beam-width = 10.

The result of using the learned rules to test against the holdout sample is shown in Table 4.1, which shows that the learned rules are quite effective in predicting and classifying. Since the inductive learning algorithm is both consistent and complete, the original positive and negative examples can be classified with perfect accuracy. Accordingly, the learned rules classify the whole set of data cases, 23 failed firms and 23 nonfailed firms, with 86.2% accuracy (Table 4.2). Compared with 83.3% accuracy resulting from the logit model used in Gentry et al. [1985], the rules generated by inductive learning appear to provide a valid decision aid for determining whether a firm has the characteristics of bankrupt firms.

5. Summary

Features such as explanation ability, heuristic inference, reasoning with uncertainty, and capabilities for incremental refinement make the knowledge-based expert system an effective tool for decision support. In this paper we have described another aspect in designing such knowledge-based decision support systems: an inductive learning method that can help automate the knowledge-acquisition process and generate decision rules.

In this paper, we have established a unified framework for problem-solving and inductive learning: both of them are treated as a process of search through state space. The problem of inductive learning can be defined in the hypothesis space and state-transitions can be achieved

through generalizations. The process of inductive learning is, thus, a sequence of generalizing initial and intermediate concept descriptions until the inductive concept descriptions are generated that satisfy the given set of examples. Just as in most of the artificial intelligence methods, inductive learning methods also suffer the problem of combinatorial explosion. We have analyzed the complexity of the methods. The empirical study shows that the algorithm incorporating beam search to expedite the search demonstrate fairly good rule-learning performance.

By using some applications in the MARBLE domain as examples, we have shown the utility of inductive inference to knowledge acquisition and to enhancing the problem-solving capability of DSSs. The incorporation of the inductive learning component enables the DSS to be adaptive in its problem-solving process, since decision rules can be generated and refined through new observations. This type of learning capability would be essential in the effort of building an intelligent DSS.

Acknowledgement

This research is supported in part by the Amoco Foundation Professorship and by a grant from the Office of Information Management sponsored by IBM. Thanks are due to Professor Michalski of the Computer Science Department for permitting us to test the AQ15 program, and to Professors Gentry and Whitford of the Finance Department for providing us the bankruptcy data used in the empirical study.

Appendix 1.

An Example of MARBLE's question-answering mode

The following is a sample question-answering session with MARBLE, a loan evaluation decision support system.

(1) "What is the customer that is applying for LOAN-1?"
**ASCEM INC.
{** indicated the user input.}

(2) "Is the customer a present customer?"
**YES

(3) "Do you want to do the extensive credit checking on this customer?"
**YES.

(4) "What is the rating of management competence?"
**F7
{F7 is a function key of showing WHY this question is needed.}

The rating of management competence is needed to determine the firm's credit-worthiness in relation to a particular loan application

RULE073
If 1) 1) the rating of management competence is high, and
2) the outside credit rating of the firm is high, and
3) credit rating based on the bank's financial analysis of the applicant's statements is high, or
2) 1) a rating of management competence is HIGH, and
2) the outside credit rating of the firm is HIGH, and
3) credit rating based on the bank's financial analysis of the applicant's statements is AVERAGE, or
3) 1) a rating of management competence is HIGH, and
2) the outside rating of the firm is GOOD, and
3) credit rating based on the bank's financial analysis of the applicant's statements is HIGH, or
4) 1) a rating of management competence is AVERAGE, and
2) the outside rating of the firm is HIGH, and
3) credit rating based on the bank's financial analysis of the applicant's statements is HIGH.
then 1) it is definite (100%) that the credit-worthiness is high.
2) it is definite (100%) that extensive credit check is finished, and
3) Inform the user of this decision.

{the user can input the value with some degree of certainty}
**HIGH
**8

(5) "What is the outside credit rating of the firm?"
**F8
{F8 is a HOW function key. The user is able to see how the value of previous parameters have been determined.}

HOW did the system determine the value of
MANAGEMENT-COMPETENCE OF LOAN-1

CUSTOMER-STATUS OF LOAN-1
CUSTOMER-NAME OF LOAN-1
{If the user choose management-competence, the following appears:}
MANAGEMENT-COMPETENCE was determined to be HIGH with certainty 80% by user input.

**HIGH

(6) What is the type of loan?
**TERM

(7) What is the firm's 3-year average net profits?
**90,000,000

(8) What is the firm's trend in net profits?
**F4
{F4 is an UNKNOWN function key}

(9) What is the industry median ratio of pretax profits to total tangible assets divided by prime rate of interest?
**2

(10) What is the firm's ratio of pretax profits to total tangible assets divided by prime rate of interest?
**2

(11) What is the firm's percentile in industry ratio of net profits to tangible net worth?
**0.8

(12) What is the firm's percentile in industry inventory turnover ratio?
**F4

(13) What is the firm's percentile in industry: net profits?
**F4

(14) If the firm's percentile ratio of pretax profits to total tangible assets exceed prime rate, answer the ratio, otherwise press F4.
**F4

(15) What is the industry median trend in net profits?
**0.2

(16) What is the ratio of cash to total current liabilities?
**0.2

(17) What is the percentile value of the firm's ratio of inventories to current assets?
**0.2

(18) What is the percentile value of the firm's current ratio?
**0.8

(19) What is the proposed loan amount?

**10,000,000

(20) What is the legal loan limit?
**120,000,000

(21) What is the maximum loan limit determined by either the firm's average deposits during the past year or during the past three years?
**100,000,000

(22) What is the ratio of proposed loan to firm's total assets?
**0.8

(23) What is the purpose of the loan?
**PLANT AND EQUIPMENT

(24) What is the proposed maturity of the loan?
**14

(25) Is the firm in a fully secured basis?
**NO

(26) THE LOAN IS GRANTED WITH CERTAINTY FACTOR = 72%

Appendix 2: Organization of MARBLE

In MARBLE, production rules are the basic form of knowledge representation. Rules can be subject to categorization in accordance with the context-types for which they are most appropriately invoked. For example, some rules deal with profitability, some with solvency, and still others deal with loan evaluation. The grammar of the rules, described by the BNF formalism, is shown in Table A.1.

To capture fully the decision rules used in business loan evaluation, MARBLE currently uses eight different context-types in its knowledge base:

LOAN	The loan application
EVALUATION	An evaluation of a new customer relationship, and
FEASIBLE	A feasibility appraisal
RECOMMEND	Detailed recommendations:
S1	The credit-worthiness of the firm in relation to the proposed loan,
S2	The indication of the extent that the customer will build the bank,
S3	The evaluation of the expected profitability to the bank of a customer relationship with the firm.
PROFITABILITY	The expected profitability of the firm:
SOLVENCY	The expected solvency ability of the firm

The context-types instantiated during the consultation session are arranged hierarchically in a data structure termed the context tree, as shown in the following figure:

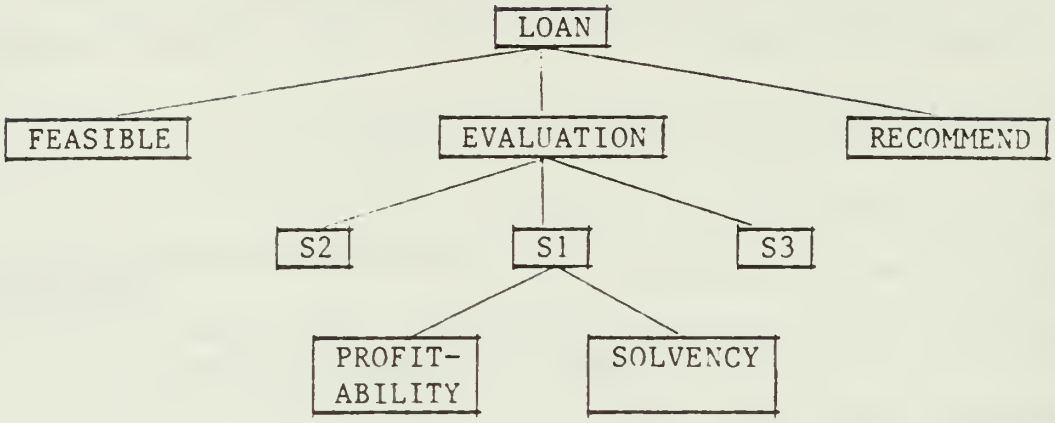


Figure A.1

The context tree helps to structure a knowledge base domain by allowing the knowledge engineer to separate a large amount of information of knowledge into logical entities. Each context can solve one part of the total problem and provide important information needed to solve the problem as a whole.

```
      <rule> :: = <premise> <action>

   <premise> :: = ($AND <condition> ... <condition>)

 <condition> :: = (<funcl> <context> <parameter>) |
                  (<func2> <context> <parameter> <value>) |
                  ($OR <condition> .. <condition>)

    <action> :: = <conclusion> | <actfunc> |
                  (DO-ALL <conclusion> ... <conclusion>) |
                  (DO-ALL <actfunc> <actfunc> <actfunc>)

<conclusion> :: = (<confunc> <context> <parameter> <value> TALLY <cf>)
```

Table A.1

Appendix 3: Inductive Inference on the Example Problem

To derive the classification decision rule for class I, we start with the first positive example, corresponding to customer A, in class I.

Step 1

--The program produces the discriminant d^{jk} of customer A against each negative examples one by one, starting with customer D in the negative example set. This process generates the following conjunctive description

[F1=H][F2=H][current-assets >$42,000][net-worth >$37,000]
[total-debt >$19,000][funds >$8,000][cash <$6,000]
[cur-liability <$55,000][inventory >$12,000]
[avg-inventory >$6,000][avg-profits >$8,000][past-acc-eval <2Y]
[account-type =C].

--Repeat the same procedure to the remaining negative examples, against customer E:

[F2=H][current-assets >$38,000][net-worth >$46,000]
[total-debt <$28,000][inventory >$14,000][avg-inventory >$6,000]
[avg-profits >$9,000] [account-type =T]

against customer F:

[F1=H][F2=H][current-assets >$52,000][net-worth >$40,000]
[total-debt <$25,000][funds >$6,000][cash <$5,000]
[cur-liability <$45,000][inventory >$11,000]
[avg-inventory >$5,000][avg-profits >$9,000]
[past-acc-eval = present][cust-status =N] [account-type =E]

against customer G:

[F1=H][F2=H][current-assets >$45,000][net-worth >$38,000]
[total-debt <$36,000][funds >$0][cash <$6,000]

[cur-liability <$57,000][inventory >$7,000]

[avg-inventory >$3,000][avg-profits >$9,000][past-acc-eval = 1Y]

[cust-status =C][account-type =C]

against customer H:

[F1=H][F2=H][current-assets >$37,000][net-worth >$29,000]

[total-debt <$27,000][funds >$7,000][cash <$6,000]

[cur-liability <$53,000][inventory >$13,000]

[avg-inventory >$5,000][avg-profits >$8,000][past-acc-eval =1Y]

[account-type =T]

against customer I:

[F1=H][F2=H][current-assets >$46,000][net-worth >$36,000]

[total-debt <$35,000][funds >$5,000][cash <$5,000]

[cur-liability <$57,000][inventory >$14,000]

[avg-inventory >$6,000][avg-profits >$0][past-acc-eval =1Y]

[account-type =C].

Generalization rules as the extension-against rule and the climbing generalization tree rule have been applied in the foregoing process in deriving these discriminants.

Step 2

--Form a set of complexes C^js by taking an attribute out of each discriminat d^{jk} generated in Step 1.

For example, by taking out the first attribute in each discriminant generated above and alternatively taking the attribute of the discriminant against customer I, d^{AI}, the following 13 complexes (C_i^js) are generated:

(1) [F1=H][F2=H][F1=H][F1=H][F1=H][F1=H],

(2) [F1=H][F2=H][F1=H][F1=H][F1=H][F2=H],

(3) [F1=H][F2=H][F1=H][F1=H][F1=H][current-assets >$46,000],

.
.
.

(13) [F1=H][F2=H][F1=H][F1=H][F1=H][account-type=C]

Another example of the complex generated in this step would be

(i) [avg-inventory >$6,000][net-worth >$46,000]

[avg-inventory >$5,000][avg-inventory >$3,000]

[net-worth >$29,000][net-worth >$36,000],

.
.
.

This process continues until all the consistent complexes are generated.

These complexes can be simplified by removing redundant components or applying generalization rules. For example, complex (3) can be simplified to [F1=H][F2=H][current-assets > $46,000] and complex (i) can be generalized to [avg-inventory $\geq$$7,000] [net-worth $\geq$$47,000] by applying the closing interval rule described in Michalski [1983] (note that the unit of input data is $1,000).

Step 3

--Choose the best complex description from the set of complexes generated in Step 2 according to the preference criterion. The preference criteria used in the example include (1) to maximize the number of positive examples covered, (2) to minimize the number of negative examples covered, and (3) to minimize the number of attributes. Then the complex selected is

[avg-inventory $\geq$\$7,000][net-worth $\geq$\$47,000],

which covers customers A, B, and C in the set of positive examples.

Step 4

--Remove the covered positive examples from the list of positive examples by the resulting description in Step 3, and apply the algorithm to the remaining positive examples. Since, in this case, all the positive examples have been covered by the single complex, the decision rule for class I is

[avg-inventory $\geq$\$7,000][net-worth $\geq$\$47,000] --> [class = I].

The same procedure produces the following decision rule for class IA:

[\$37,000 $\leq$ net-worth $\leq$ \$48,000][inventory >\$8,000]

--> [class = IA];

and for class II

[Fl=H,A][total-debt $\geq$ \$26,000]

--> [class = II].

For the given set of training examples, the three classification rules thus generated covered all the positive examples but none of the negative examples, i.e., the induction process is both complete and consistent. These decision rules not only can be used for credit classification, each of the rules is also a description of a "concept" learned from observing the classification examples. The set of decision rules generated by the inductive learning program can then be added to the MARBLE system.

Appendix 4: Attributes of the Training Examples Used in the Empirical Study

In the empirical study, we apply the Inductive Inference Algorithm to the problem of bankrupty prediciton. To indentify the relevant attributes for learning the characteristics (concepts) of bankrupt firms, we adopted the cash-based funds flow components developed in Gentry et al. [1985]. The funds flow components involved are funds from operations (NOFF), working capital (NWOFF), financial (NFFFF), fixed coverage expenses (FCE), capital expenditures (NIFF), dividends (DIV), and other asset and liability flows (NOW & LF).

The ratio of these components to the total net flow (TNF) form the first seven attribute of each example. The eighth attribute is a scale measure, calculated by total net flows/total assets (TNF/TA). Thus, each training example consists of the following eight attributes (1) NOFF/TNF, (2) NWCCFF/TNF, (3) NOA & LF/TNF, (4) NFFF/TNF, (5) FCE/TNF, (6) NIFF/TNF, (7) DIV/TNF, and (8) TNF/TA.

REFERENCES

Alter, S. L., 1980, Decision Support Systems: Current Practice and Continuing Challenges, (Addison-Wesley: Menlo Park, CA).

Anderson, J. R., 1983, The Architecture of Cognition, (Harvard University Press: Cambridge, MA).

Angluin, D. and Smith, C., 1983, "Inductive Inference: Theory and Methods," Computing Surveys, Vol. 15, No. 3, pp. 237-269.

Bonczek, R. H., Holsapple, C. W. and Whinston, A. B., 1980, "Future Directions for Developing Decision Support Systems," Decision Science, Vol. 11, pp. 616-31.

_____, 1981, "Representing Modeling Knowledge with First Order Predicate Calculus," Operations Research.

_____, 1983, "Specification of Modeling and Knowledge in Decision Support Systems," in H. G. Sol (ed.), Processes and Tools for Decision Support, (North Holland: Amsterdam).

Buchanan, G. B. and Mitchell, T. M., 1978, "Model-Directed Learning of Production Rules," Pattern-Directed Inference Systems, in Waterman, D., and Hayes-Roth, F. (eds.) (New York: Academic Press).

Buchanan, B. and Shortliffe, E., 1984, Rule-Based Expert Systems, (Addison-Wesley: Reading, MA).

Cohen, K. J., Gilmore, T. C., and Singer, F. A., 1966, "Bank Procedures for Analyzing Business Loan Applications," Cohen, K. J., Hammer, F. S. (eds), in Analytical Methods in Banking, pp. 219-49.

Davis, R., 1979, "Interactive Transfer of Expertise: Acquisition of New Inference Rules," Artificial Intelligence, Vol. 12, pp. 121-57.

_____, Buchanan, B. and Shortliffe, E., 1977, "Production Rules as a Representation for a Knowledge-Based Consultation Program," Artificial Intelligence, pp. 15-45.

Dietterich, T. G., Lonclon, R., Clarkson, K., and Dromey, R., 1981, "Learning and Inductive Inference," D. Cohen and E. Feigenbaum (eds) in Handbook of Artificial Intelligence, (Kaufman: Los Altos), pp. 323-525.

Dolk, D. and Konsynski, B., 1984, "Knowledge Representation for Model Management Systems," IEEE Trans. on Software Engineering, Vol. SE-10, No. 4, pp. 619-627.

Dutta, A. and Basu, A., 1984, "An Artificial Intelligence Approach to Model Management in Decision Support Systems," Computer, Vol. 17, No. 9, pp. 89-98.

Elam, J. J. and Henderson, J. C., 1980, "Knowledge Engineering Concepts for Decision Support System Design and Implementation," Proceedings of Fourteenth Annual Hawaii International Conference on System Sciences, (Western Periodicals: North Hollywood, CA).

Gentry, J., Newbold, P., and Whitford, D., 1985, "Classifying Bankrupt Firms with Funds Flow Components," Journal of Accounting Research, Vol. 23, No. 1, pp. 146-160.

Goul, M., Shane, B., and Tonge, F., 1986, "Knowledge-based DSS in Strategic Planning Decisions," Journal of Management Information Systems, Vol. II, No. 4.

Haslem, J. A. and Longbrake, W. A., 1972, "A Credit Scoring Model for Commercial Loans, A Comment," Journal of Money, Credit and Banking, Vol. , pp. 733-34.

Kaplan, R. S. and Dietrich, J. R., 1982, "Empirical Analysis of the Commercial Loan Classification Decision," Accounting Review, Vol. 58, No. 1, pp. 18-38.

Lee, W. and Ray, S., 1986, "Probabilistic Rule Generator," Report No. UIOCDCS-R-86-1263, (Department of Computer Science: Champaign, IL).

Lenat, D. B., 1976, "AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search," Ph.D. dissertation, Stanford University.

Lowerre, B., 1976, "The HARPY Speech Recognition System," Ph.D. Dissertation, (Department of Computer Science, Carnegie-Mellon University: Pittsburg, PA).

Michalski, R. S., 1980, "Pattern Recognition as Rule-Guided Inductive Inference," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 2, pp. 249-361.

_____, 1983, "A Theory and Methodology of Inductive Learning," in Michalski, R., Carbonell, J., and Mitchell, T., (eds.), Machine Learning, (Tioga: Palo Alto).

_____, and Larson, J., 1978, "Selection of Most Representative Training Examples and Incremental Generation of VL_1 Hypotheses: The Underlying Methodology and the Description of Programs ESEL and AQ11," UIUCDCS-R-78-867, (Department of Computer Science, University of Illinois: Champaign, IL.

_____ and Chilausky, R. L., 1980, "Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an

Expert System for Soybean Disease Diagnosis," Policy Analysis and Information Systems, Vol. 4, No. 2, pp. 125-60.

_____, and Stepp, R., 1983, "Learning from Observation: Conceptual Clustering," in Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. (eds), Machine Learning, (Tioga: Palo Alto).

Mitchell, T., 1982, "Generalization as Search," Artificial Intelligence, Vol. 18, pp. 203-26.

Newell, A., 1973, "Production Systems: Models of Control Structures," in W. G. Chase (ed.), Visual Information Processing, (Academic Press: New York).

_____, and Simon, H. A., 1972, Human Problem Solving, (Prentice-Hall: Englewood Cliffs).

Orgler, Y. E., 1970, "A Credit Scoring Model for Commercial Loans," Journal of Money, Credit and Banking, pp. 435-45.

Ow, P. S., 1984, Heuristic Knowledge and Search for Scehduling, Ph.D. Thesis, (The Graduate School of Industrial Administration, Carnegie-Mellon University: Pittsburg, PA).

Quinlan, J. R., 1979, "Discovering Rules from Large Collection of Examples: A Case Study," in Michie, D. (ed.) Expert Systems in the Micro Electronic Age, (University Press: Edenburgh).

Rendell, L., 1983, "A New Basis for State-Space Learning Systems and a Successful Implementation," Artificial Intelligence, Vol. 20, pp. 369-92.

Shaw, M., 1985, "A Knowledge-Based Framework for Distributed Decision Support," Simulation and Modeling, Vol. 16, in W. Vogt and M. Mickle, (eds), (The Instrument Society of America).

_____, 1986, "Applying Inductive Learning to Enhance Knowledge-based Expert Systems," Faculty Working Paper No. 1300, (Department of Business Administration, University of Illinois: Champaign); Decision Support Systems (forthcoming).

_____ and Tu, P. L., 1986, "An Integrated Framework for Knowledge-Based Decision Support," Working Paper, University of Illinois, Department of Business Administration.

Sprague, R. H. and Carson, E. D. 1982, Building Effective Decision Support Systems, (Prentice-Hall Inc.: Englewood Cliffs, NJ).

Stohr, E. A., 1986, "Decision Support and Knowledge-based System: A Special Issue," Journal of Management Information Systems, Vol. II, No. 4.

Vere, S. A., 1975, "Induction of Concepts in the Predicate Calculus," Fourth International Joint Conference on Artificial Intelligence, Tbilisi, USSR, p. 281-87.

Winston, P. H., 1975, "Learning Structural Descriptions From Examples," in The Psychology of Computer Version, (McGraw-Hill: New York).